

# Test Scenario Generation for Web Application

Abhay Kumar Vijay

111CS0163



Department of Computer Science and Engineering

National Institute of Technology, Rourkela-769008

May-2015

# Test Scenario Generation for Web Application

A Thesis submitted in partial fulfilment of the requirements for  
the degree of

Bachelor of Technology in Computer Science and Engineering

By

Abhay Kumar Vijay

(111CS0163)

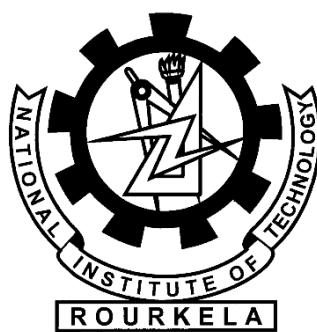
Under the Supervision of

Prof. D. P. Mohapatra

Associate Professor,

Department of Computer Science & Engineering

NIT Rourkela



Department of Computer Science and Engineering  
National Institute of Technology,  
Rourkela-769008  
May-2015



## Certificate

This is to certify that the work in the thesis entitled “Test **Scenario Generation for Web-Application**” submitted by **Abhay Kumar Vijay (111CS0163)** in partial fulfilment of the requirements for the award of Bachelor of Technology Degree in Computer Science & Engineering at NIT Rourkela is an authentic work carried out by him under my guidance.

To the best of my knowledge the matter embodied in the project has not been submitted to any University/Institute for the award of any B.Tech degree.

**Prof. D. P. Mohapatra**

Associate Professor,

Dept. of Computer Science & Engg.

NIT Rourkela.

# Declaration

I hereby declare that all the work contained in this report is my own work unless otherwise acknowledged. Also, all of my work has not been previously submitted for any academic degree. All sources of quoted information have been acknowledged by means of appropriate references.

Abhay Kumar Vijay  
NIT Rourkela

# Acknowledgement

The project work would not have been finished without specifying those who have provided their direction. I would like to express my earnest gratitude to Prof. D. P. Mohapatra for his consistent guidance and direction throughout my project work. As my supervisor, he has consistently kept me motivated to stay focused towards achieving the goal. His observations and wisdom helped me to establish an overall direction for the research and to make an in-depth study. Without his useful advice and assistance it would not have been possible for me to complete this thesis. I am very much grateful to the faculty members of Department of Computer Science and Engineering, National Institute of Technology, Rourkela.

Abhay Kumar Vijay

# Abstract

Software testing is one of the most imperative part of software development life cycle. Generation of test sequences which is crucial for software testing is one of the challenging task. Testing can be possible either manually or automatically. Automated testing reduces cost as well as time so it is better than manual testing. Control flow graph are used to show the workflow of dynamic and behavioural aspects of the software system. Test sequences can be generated using the control flow graph. A test model is described to represent the data flow information of JSP pages with considering of data variables in it. We have developed a tool for automatic generation of control flow graph from JSP pages which represent the data flow information in it. We have generated test sequence from control flow graph using this tool which can be useful for finding data anomalies in JSP pages.

# Table of Contents

Chapter 1	Introduction	1
1.1	Motivation.....	2
1.2	Objective .....	3
1.3	Organization of Thesis.....	3
Chapter 2	Basic Concepts.....	4
2.1	Java Server Pages (JSP) .....	4
2.2	Control flow graph (CFG) .....	5
2.3	Summary.....	5
Chapter 3	Related Work	6
3.1	Web Application Testing.....	6
3.2	Data flow model of Web Application.....	6
Chapter 4	Proposed Work	8
4.1	Data flow test artefacts in JSP .....	8
4.2	Data flow test model.....	9
4.3	Generation of test scenario from CFG.....	9
4.3.1	Extracting Nodes from JSP pages.....	10
4.3.2	Computation of definition-use chain.....	11
4.4	Summary.....	12
Chapter 5	Implementations and Results	13
5.1	Tools used .....	13
5.1.1	Eclipse.....	13
5.1.2	Graphviz.....	14
5.2	Screenshots.....	15
5.2.1	Example Login JSP page.....	15
5.2.2	Example Multiple JSP pages .....	18
5.3	Summary.....	20
Chapter 6	Conclusion and Future Work	21
	Conclusion.....	21
	Future Work .....	21

# List of Figures

Figure 4.1 Example of data interactions between JSP pages.....	10
Figure 5.1 Sample JSP login page .....	15
Figure 5.2 CFG of JSP login page .....	16
Figure 5.3 def-use chain from Figure 5.2.....	17
Figure 5.4 Multiple JSP Pages .....	18
Figure 5.5 CFG of Multiple JSP Pages .....	19
Figure 5.6 def-use chain from Figure 5.5.....	20



# Chapter 1

## Introduction

The basic objective of software testing is delivering software with higher quality. Software testing is conducted to discover the faults in software which causes its fail. But also it is time consuming and costly to perform. Software testing can be either manual or programmed. Automated testing is generally done using software testing tools. Automated software testing is more productive than manual testing as it decreases time and expense sum.

Verity of software testing prompts to standardization. Some significant software testing procedure includes black box testing and white box testing. Fundamentally software testing is a technique to approve and check the program whether it meets the client's need or not.

Functional testing of software known as black box testing. It does not check how the software is implemented. Functionality of a software is ensured by black box testing (what it does) without having any insight into the internal structure of the software.

White box testing tests all conceivable way that are accessible in a software. It checks working of a software and the internal structure. It confirm the quality of the source code.

Java server pages (JSP) pages have been used in Web applications based on Java to handle web requests, to interact with Java components like to generate dynamic pages, and Java beans. It is important to guarantee that the JSP pages are programmed correctly and their

interactions with other components like SQL are handled properly. However, In order to generate dynamic pages JSP pages usually mix up scripts (i.e., JSP servlets) with HTML statements.

Java Server pages as a programming language don't have compile time testing and, subsequently, can be left anomalies. There are more test process for JSP web-application testing, such as Cactus and HTTP Unit, Testing of JSP application is viewed as troublesome and test cases are still simply designed. Most imperative, JSP application have presented XML like tags, variables and objects [3]. The implicit objects and variables can raise some problem when the inside of JSP application program is practiced using general data flow testing methods.

## 1.1 Motivation

Software testing takes a lot of time and exercise of the whole software development process. As the size of the software product increases so does its complexity which increases time taken for testing and debugging and the maintenance cost of the product. It is very hard to make a test sequence that can test all the deformity of the software on one try The input database of a software is so large. Proper testing requires detection of error and fault in the software. Test sequences can be utilized to identify those errors. Generation of test sequences selection of the best test sequence out of them needs legitimate procedure.

Generating all test sequences may not be a decent approach. Since as the size and complexity of the software increases, number of test sequences generated from it also increases. So we need a technique to limit those test sequences so that it can reduce both time and expense for testing a software product.

## 1.2 Objective

The objective of this project is to parse the Java Server Pages into a program and generate its control flow graph and then test sequences out of it. We will analyse the possible data flow test artefacts introduced by the JSP pages. A test model will be proposed to abstract the data flow information of various JSP variables and implicit objects. A methodology for computing the data flow test paths including the variables will be described and illustrated. From those, test sequences having higher priority can be chosen for software testing.

## 1.3 Organization of Thesis

Chapter 1 discusses a brief introduction to JSP web application and their types. It also emphasizes on the need for data flow testing and test sequence generation. Chapter 2 discusses two main concept which are used in our work. The main function of JSP and control flow graph will be given. Chapter 3 gives a brief account of the related work by various authors. In chapter 4 first part of proposed work is generating control flow graph from JSP page and second part is generating test sequence from control flow graph. We will discuss our method for generating control flow graph further more we will discuss an algorithm for generate the test sequence from control flow graph. Chapter 5 discusses the tools we have used for our work. The next part showing two example we taken as our input and generated control flow graph and then generated test sequence for both of the example is showing. These are the output of our program which is showing our proposed work. Chapter 7 concludes the thesis and gives a brief account of the future work that can be continued from where we have left.

# Chapter 2

## Basic Concepts

In this section we discuss some basic concepts and definitions.

A Test sequence a software is of a path generated from the control flow graph. It consists of a number of nodes. We traverse the control flow graph from Start node to Exit node, nodes encountered during that traversing are stored in some array. We reach the end point using our algorithm, then the array of nodes is printed as a test sequence. The test sequence gives the way on which testing should be done for the object. The software will be tested more efficiently if we choose the best test sequence from it so.

### 2.1 Java Server Pages (JSP)

Java Server Pages (JSP) is a server-side programming technology that enables the creation of dynamically generated web pages based on XML, HTML, or other document types. JSP have access to the entire family of Java APIs, including the JDBC API to access enterprise databases. JSPs are usually used to deliver XML and HTML pages through the use of OutputStream. They can deliver other types of data as well [4].

The Web container makes implicit objects like application, Exception, session, servletContext, pageContext, request & response implicit objects.

JSP pages use few delimiters for scripting capacities. The most essential is `<% ... %>`, which encloses a JSP code or scriptlet. When the user request the page fragment of Java code within scriptlet runs. Other common delimiters in JSP include `<%-- ... %>` for comments, `<%= ... %>` for expressions.

Java code is not needed to be finished or independent within its scriptlet element block, however can straddle text content providing the page is syntactically correct. For example, any Java loop blocks opened in one scriptlet element must be correctly closed in same page for successfully compilation. Text which falls into this type of split block of code is subject to that code, so text inside an if block will only appear in the output when the if condition evaluates to true; likewise, text inside a loop construct may seem numerous times in the output relying on how frequently the loop body runs.

## 2.2 Control flow graph

A control flow graph (CFG) in software engineering is a representation of all paths that might be traversed through a program using graph notation.

Here in a control flow graph each node in the graph represents a straight-line piece of code, which is a basic block, jump targets start a block, and jumps end a block. The control flow is represented using directed edges. A control flow graph shows how events in the program are sequenced. There is two special nodes in control flow graph, one is entry node, through which control starts and another is exit node through which control ends.

## 2.3 Summary

Here we discussed two main concept which are used in our work. First is JSP (Java Server Page) and Control flow graph. We have summarize that how they were used in our work. The main function of JSP and control flow graph is given. These terminology are important for our work.

# Chapter 3

## Related Work

In this section, we present a brief survey of the existing literatures those are closely related to our work.

### 3.1 Web Application Testing

In our work. recent time, many approaches for testing web application are discussed. Yang [2] extend customary software testing architecture to support Web application testing. A set of tools is developed to help develop test cases, analyse documents, execute tests, support test measurement and screen the failures. Kung et al. [8] present a test model that edited the unstructured Web pages in terms of relationships and objects. They consider the web documents as objects and analyze their possible control flow interactions with other segments of the Web applications. A data flow test system is there to select test paths for analysing scripts from inter-object, intra-object, and inter-client point of view.

### 3.2 Data Flow Model of Web Application

Ricca and Tonella [6] depict a proposed model that demonstrate the Web application, frames, and their interactions in the applications. The model executes both static web application and dynamic Web application web pages. From this work, test cases can be derived to test the control flow of data among the Web pages. Lucca [11] suggested intricate test model to represent various elements of a Web application. In view of their model, a procedure is proposed to produce preliminary test cases for unit and integration testing of Web applications.

In view of the client data flow, It is proposed that the testing as data flow of HTML pages and conceivable creations of the very small parts. Offutt [12] present a way to deal Web software by avoiding the server side output acceptance. They side pass the data validation from the input parameter, input value and data flow viewpoints to test the anomalies of Web software. The experimental results are introduced to perform the usefulness of the described methodology.

# Chapter 4

## Proposed Work

Our project consists of some existing work as well as some proposed work. Chien-Hung Liu gave an insight about how to extract nodes from JSP pages without using any intermediate model. Node representation of the JSP file is used to extract information about the diagram. We used control flow representation from the corresponding activity diagram.

### 4.1 Data flow test nodes of JSP pages

Testing of JSP page mostly concentrates on the defined variables and their important uses for finding the data errors of software. The variable can be divided into two parts by their use p-use (process use) or c-use (computation use). A p-use happens whenever a variable is used in a predicate line of code and a c-use happens whatever point an object is utilized in a computation or output line of code. Test chains of a JSP page are selected based on the definition- use (or def-use) chains of variables. The path from the definition to the use of the object, without any redefinition or use, this path is called def-use chains.

To find the data errors in JSP applications, we need to consider variables of the JSP pages, the action tags and variables introduced by the Java Server Pages. The variables, example response and request, permit JSP designers to get JSP provided services and resources without expressly declaring the objects. Specifically, some of the implicit objects have the function to control the flow and the input/output of JSP pages. For instance, the response object can be used to access the data supplied in client requests. The previous response variable can be access to all JSP pages.



## 4.2 Data flow test model

A JSP page can contain huge amount of functions. The control structure of JSP application are not like the class and object type, it procedural. The function inside a JSP page can accessed within the page only from the beginning of entry point. So, for extracting the control flow of data in JSP application we consider the functions and find their data flow to other functions.

JSP pages do not have an explicit calling-hierarchy and the calling functions in JSP pages can be dependent of flow of the data within the page unlike the traditional program. In specifically, through accessing the particular object in a page, various data flow scenarios can lead the various data flow interaction in the whole JSP pages. Such data flow interaction can be found as the user uses these chains, though it happen on the control flow of data in JSP application. To find data flow artefacts explicitly, we also consider the control flow of data introduced by the functional variables.

## 4.3 Generation of test scenario from CFG

If we only see the functionality, JSP pages are not like object class rather they are like scripting pages though a JSP page can have several procedures functions. The procedure and methods declared in a JSP page can only be called within that JSP page because it has only a starting point. So, for the information of control flow of data in JSP pages, we did not conclude JSP pages as scripting but functional and capture their intra-procedural and inte-procedural control flow of data objects.

### 4.3.1 Extracting Nodes from JSP page:

The `sendRedirect()` function of the request object allows a web response to be send to another JSP page. Data flow in the JSP pages can be affected by this and it can occurs data transfer among the different JSP pages and Java beans or between two JSP pages. The data of variables `var2` and `var1` can be passed from `page2.jsp` to `page1.jsp` through the response method, which represents data transfer between `page2.jsp` and `page1.jsp` pages. In the JSP based application there several a set of tags, such as `<jsp:include>` and `<jsp:param>` in addition to the implicit objects like `XML`. JSP pages, to transfer the from one JSP page to another page and Java objects, uses action tags that are handled by with code handlers. So there is data interaction among these pages. For example, consider the `login.jsp` page, in Line 6, the `login.jsp` page have variable `test` which uses `<jsp:useBean>`. Furthermore as variable reference to the `TestBean` variable, this `test` variable is then in line 9, used. The `test.verify()` function is used in the passing data of variables `passwd` and `login` from the `login.jsp` page.

```
//part of page1.jsp
1 String var1 = "my_value1"
2 String var2 = "my_value2"
3 response.sendRedirect("page2.jsp?param1=" + var1 + "&param2=" + var2);



---


//part of page2.jsp
4 String my_var1 = request.getParameter("param1");
5 String my_var2 = request.getParameter("param2");
```

Figure 4.1 Example of data interactions between JSP pages.

Data variables def(var1,1) and def(var2,2)

Data variables c-use(param1,4) and c-use(param1,4)

sendRedirect() method of response object.

Page2.jsp:

Data variables def(my\_var1,4) and def(my\_var2,5).

Data variables c-use(param1,4) and c-use(param1,4)

### 4.3.2 Computation of definition-use chains:

If we are finding the data anomalies considering the objects in JSP pages, the control flow is traditional, we have defined the “use” and “definition” nodes for Java variables or objects. The node attribute where the variable or object is used would be use node. Similarly node attribute where the variable or object is defined would be def node. Path generated using control flow graph.

#### **Algorithm:**

1. Set current node i = “Entry”.  
Set visited status for every node to 0.
2. Traverse from the current node(i)  
Update visited node status for current node from 0 to 1.
3. If the current node is “def(x,i)” node where x is object  
If “def(x,i)” previously processed than goto 5  
If “def(x,i)” is new than goto 4  
Else set current node “i” to next unvisited node.
4. The values of current node is “i” and object is “x”

- a. Set the value of current node  $j = i$  and traverse the graph
- b. If the current node is not visited

Check whether it is “c-use( $x_j$ )”

- c. If true than print the value of node between node  $i$  and  $j$

Break the traversal and goto 5.

If false than traverse next node until the last one.

5. Set the value of current node “ $i$ ”

Repeat the step 3 until every node is traversed.

## 4.4 Summary

In this chapter we have discussed our proposed work over the existing work. First part of proposed work is generating control flow graph from JSP page and second part is generating test sequence from control flow graph. We have discussed our method for generating control flow graph further more we have discussed an algorithm for generate the test sequence from control flow graph.

# Chapter 5

## Implementations and Results

Here we implement all methods proposed earlier with java program using Eclipse IDE. We take JSP as an input to the program and following representation of the activity diagram using Graphviz.

### 5.1 Tools used

We use the following tools in order to code and implement the programs and finally to get the desired test sequences.

- Eclipse IDE
- Graphviz

#### 5.1.1 Eclipse

Eclipse is a platform that has been developed for building integrated web and application development tooling. Eclipse is an integrated development environment (IDE). The IDE is called Eclipse PDT for PHP, Eclipse JDT for Java and Eclipse CDT for C. The Eclipse is a software which provides the base for the Eclipse IDE is collection of plug-ins and is designed to be more using extra plug-ins. Developed using Java, the Eclipse platform can be used to develop rich client applications, integrated development environments and other tools. The most important function of Eclipse is its plug-in system. We can download different plug-in tools into the eclipse environment and can be used them in the project. We have to develop and API to convert the java code to dot language which runs on Graphviz.

### 5.1.2 Graphviz

Graphviz (Graph Visualization Software) is a software of open-source tools for drawing graphs specified in DOT language scripts. It also gives libraries for software applications to use the tools. Graphviz is free software licensed under the Eclipse Public License. We have used this software in our project to visualize control flow graph in a better way that is in the form of a control flow graph instead of an adjacency list or adjacency matrix. The output of the program is converted into DOT language that is read by Graphviz and is written into an output file in the same format. Graphviz reads from the output file to generate the control flow graph that the user can visualize. Graphviz uses dot language to represent the graph.

## 5.2 Screenshots

### 5.2.1 Sample Login JSP page

```
1 <%@ page info = "example1.jsp" %>
2 <%@ page language= "java" import "java.util.*" %>
3 <%@ page import "examples.*" %>
4 <html>
5 <body>
6 <jsp:useBean id= "test" scope="session" class "examples.TestBean" />
  <%
7 String login= request.getParameter("login");
8 String passwd= request.getParameter("passwd");
9 int num = test.verify(login,passwd);
10 if(num==0){
    %>
11     UserId:<jsp:getProperty name="test" property="userId"/>
12     <a href="next.html">Click Here to Continue</a>
    <%
    }else{
    %>
13 <jsp:forward page="loginMsg.jsp" flush="true">
14 <jsp:param name="my_num" value="<%= num %>" />
</jsp:forward>
<%
}
%>
15 </body>
16 </html>
```

Figure 5.1 Login page in JSP

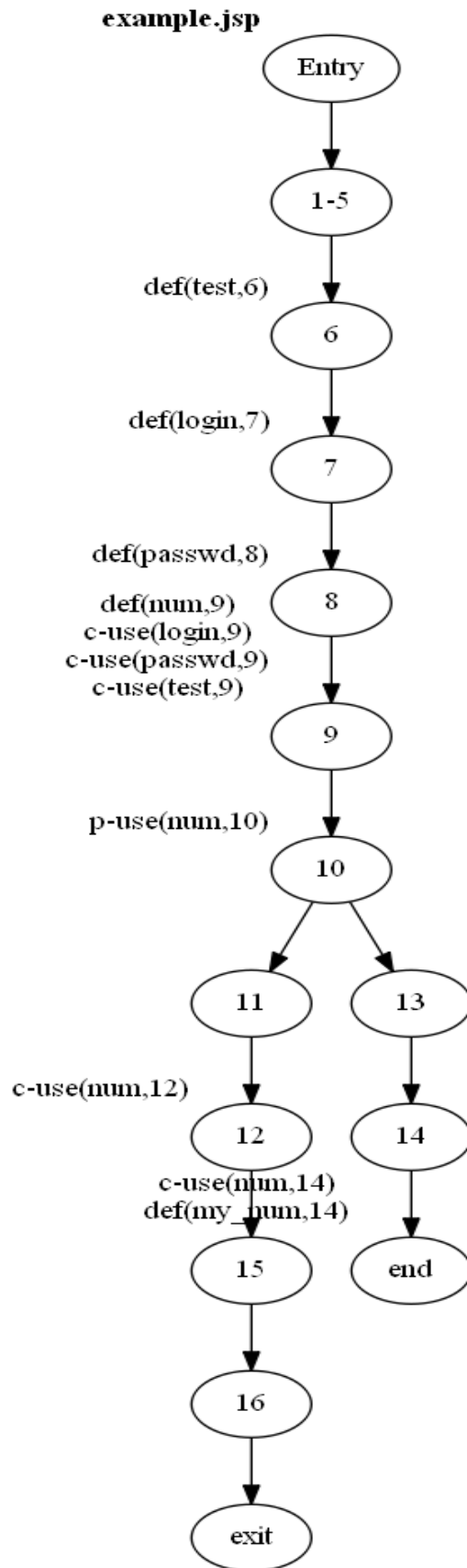


Figure 5.2 CFG Diagram of JSP



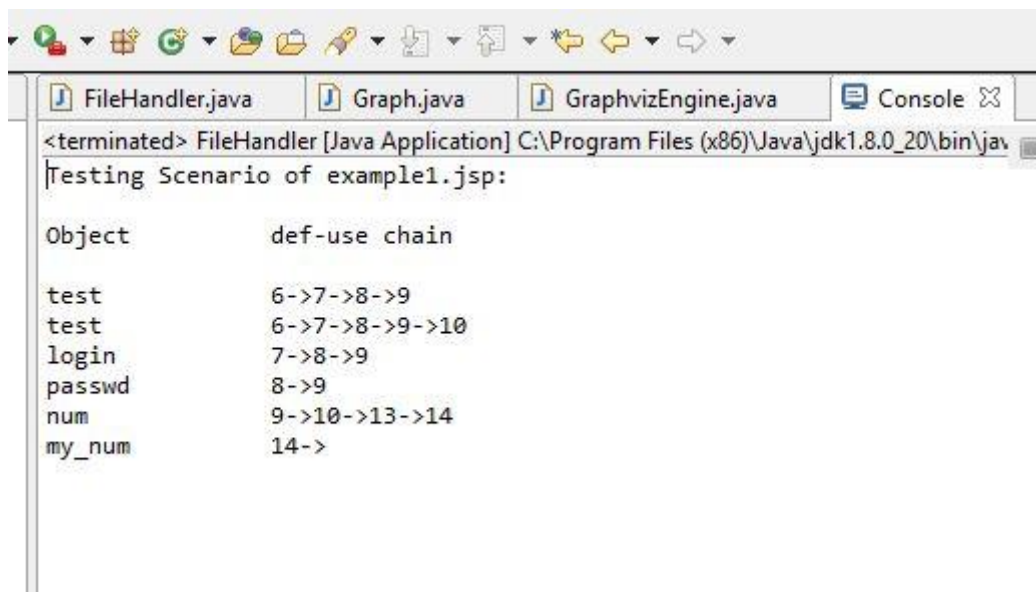


Figure 5.3 def-use chain from the Figure 5.2 CFG

## 5.2.2 Multiple JSP pages

```
<%@ page info = "icfg.jsp" %>
<%@ page language="java" import "java.util.*" %>
<%!
1      int y=0;
      public void inc(int x){
2          y=x+1; }
3 %><html><body>
4 String x=(String)request.getParameter("user_x");
5 if(Integer.parseInt(x)>0){
6     inc(Integer.parseInt(x));
7 %>
8 <jsp:forward page="fw.jsp">
9   <jsp:param name="my_y" value="<%= y %>" />
10  </jsp:forward
11  <%} else {%>
12 <jsp:include page= "in.jsp" flush="true" />
13 <% } %>
14 icfg.x = <%= x %></br>
15 </body></html>

```

---

```
<%@ page info = "fw.jsp" %>
<%
12 String x=(String)request.getParameter("user_x");
13 String y=(String)request.getParameter("my_x");
14 %>
15 fw.x=<%= x %><br>
16 fw.y=<%= y %>
17 </body></html>

```

---

```
<%@ page info = "in.jsp" %>
<%
17 String x=(String)request.getParameter("user_x");
18 %>
19 in.x = <%= x %> <br>

```

Figure 5.4 multiple pages in JSP

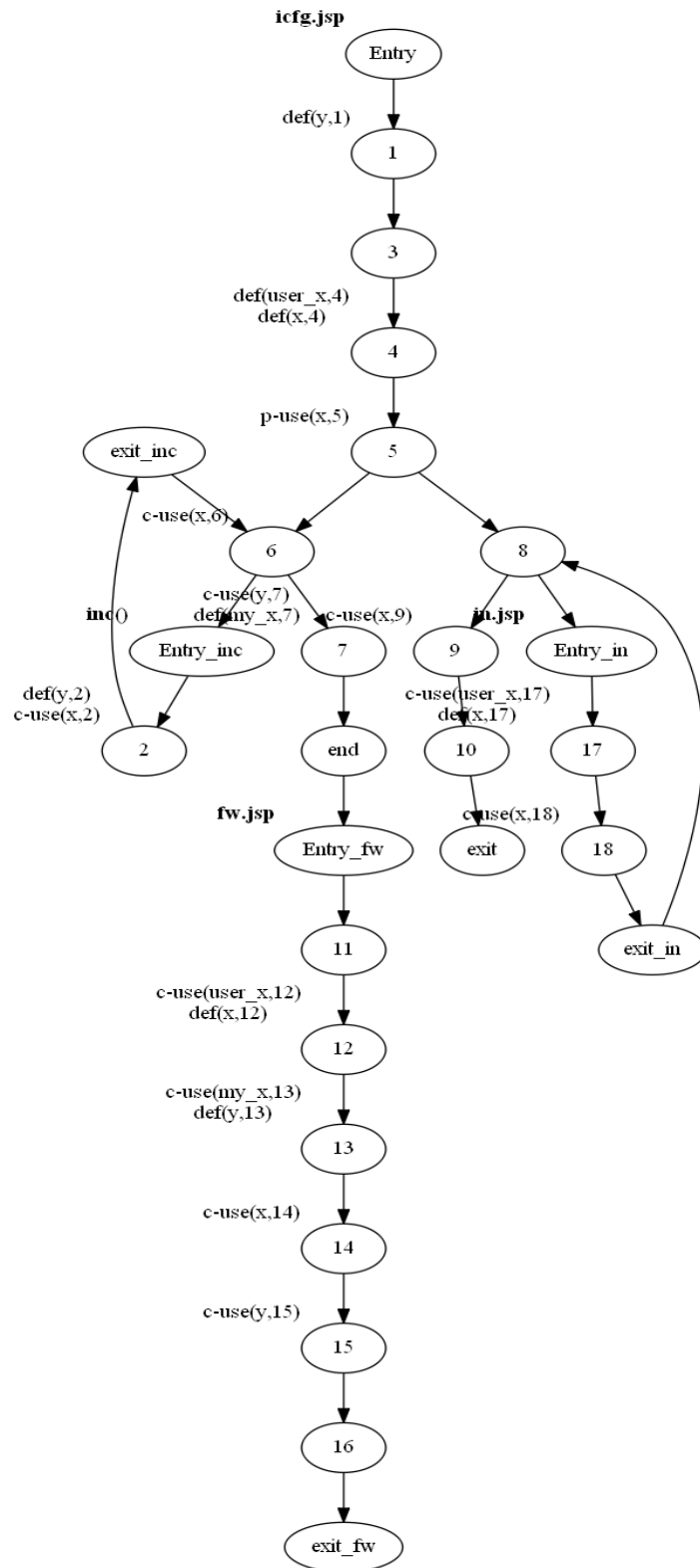


Figure 5.5 CFG Diagram of JSP pages

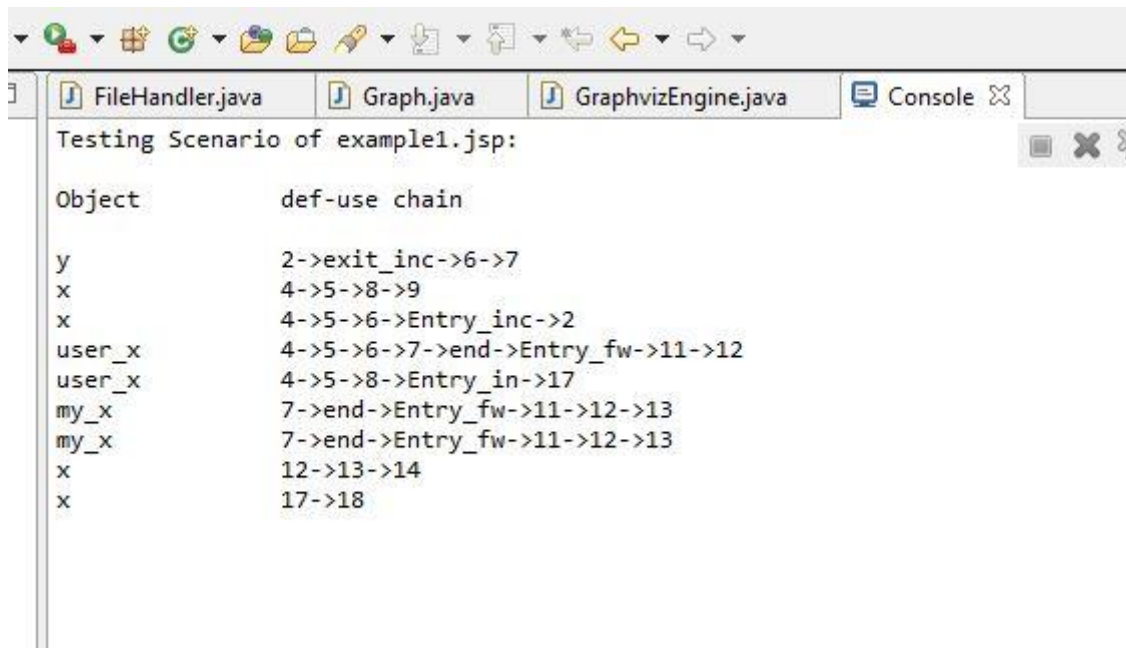


Figure 5.6 def-use chain from Figure 5.5 CFG

### 5.3 Summary

In the chapter we have discussed the tools we have used for our work with how they were useful. The next part showing two example we taken as our input and generated control flow graph and then generated test sequence for both of the example is showing. These are the output of our program which is showing our proposed work.

# Chapter 6

## Conclusion and Future Work

### 6.1 Conclusion

This thesis has introduced a method to control flow of data analysing and JSP-based web applications testing. A method introduced to represent the JSP pages into control flow graph. It can also be referred as direct control flow graph. From this JSP representation, we are able to extract corresponding nodes and edges between them. Using those we have generated test sequences on this CFG. Test models are proposed to represent the JSP control flow of data test objects pages with the considerations of the control flow characteristics of various JSP action tags and implicit objects.

### 6.2 Future Work

Different JSP applications will be taken as input to the program and its test sequences will be generated. Future work include an experimental study to assess the introduced methodology for JSP based applications and an analysing to incorporate the control flow of data test objects of JSP pages for supporting the data testing and analysis of JSP pages.

# Reference

- [1] Data flow analysis and testing of JSP-based Web applications, Chien-Hung Liu, Department of Computer Science and Info. Engg, National Taipei University of Technology.
- [2] J.-T. Yang, J.-L. Huang, F.-J. Wang, W.C. Chu, Constructing an object-oriented architecture for web application testing.
- [3] H. Bergsten, JavaServer Pages, second ed., O'Reilly & Associates.
- [4] S. Elbaum, S. Karre, and G. Rothermel, Improving web application testing with user session data.
- [5] A. Ginige, S. Murugesan, Web engineering: an introduction.
- [6] F. Ricca, P. Tonella, Analysis and testing of web applications, Proceedings of the International Software Engineering Conference
- [7] C. Kallepalli, J. Tian, Measuring and modeling usage and reliability for statistical web testing.
- [8] D. Kung, C.-H. Liu, P. Hsia, An object-oriented test model for testing web applications, in: Proceedings of Computer Software and Applications Conference.
- [9] C.-H. Liu, Data flow analysis and testing of java server page, in: Proceedings of the First Workshop of Quality Assurance and Testing Web-Based Applications.
- [10] C.-H. Liu, D. Kung, P. Hsia, C.-T. Hsu, Structural testing of web applications, in: Proceedings of International Symposium on Software Reliability Engg.

- [11] G.A.D. Lucca, A.R. Fasolino, F. Faralli, U.D. Carlini, Testing web applications, in: Proceedings of International Conference on Software Maintenance.
- [12] J. Offutt, Y. Wu, X. Du, H. Huang, Bypass testing of web applications, in: Proceedings of the Fifteenth IEEE International Symposium on Software Reliability Engineering.